Is QUIC Quicker with HTTP/3? An Empirical Analysis of Quality of Experience with DASH Video Streaming

Sindhu Chellappa and Radim Bartos
Department of Computer Science
University of New Hampshire
Durham, USA
{sindhu.chellappa, radim.bartos}@unh.edu

***Abstract*—Video streaming contributes to significant traffic on the internet. MPEG-DASH is a standard method to deliver video using short segments served by a content delivery network over HTTP. The Adaptive Bit Rate (ABR) algorithms used in MPEG-DASH play a vital role in delivering better Quality of Experience (QoE) to the users. The ABR algorithms are known to perform well for HTTP/2 over TCP. To address the need for faster data delivery and lower connection establishment, a transport layer protocol QUIC and an application layer protocol HTTP/3 are being widely deployed. With this transition, it is highly desirable to revisit the ABR algorithms in terms of QoE they deliver. In this paper, the QoE/performance of the ABR algorithms are examined and compared between HTTP/3 over QUIC and HTTP/2 over TCP. Our results show that among the existing ABR algorithms, low latency streaming algorithms work in favor of HTTP/3 over QUIC and outperforms HTTP/2 over TCP in many scenarios. In lossy network conditions, HTTP/3 over QUIC achieves higher QoE by taking advantage of higher throughput of QUIC and downloading media segments faster than HTTP/2 over TCP.**

***Index Terms*—HTTP/3, QUIC, low latency streaming, QoE, DASH, ABR**

I. Introduction

Video streaming accounts for about 84% traffic on the internet [1]. Dynamic Adaptive Streaming over HTTP (DASH) is used by leading streaming services like YouTube, Netflix, Cloudflare and AWS to deliver on-demand or live streaming media. ABR algorithms attempt to optimize the bit rate selection to maximize the Quality of Experience (QoE) depending on the current network and playback conditions.

In DASH, the data is carried over the application protocol, HTTP (HyperText Transfer Protocol). In 2015, Internet Engineering Task Force (IETF) standardized HTTP/2. To handle more data within the same connection, HTTP/2 supports multiple streams. In the protocol stack, HTTP/2 operates on top of the transport protocol TCP (Transmission Control Protocol). TCP sees the data transported as a stream of bytes and it does not know it is transporting HTTP. When the packets are delayed or lost, the entire TCP connection has to wait, to ensure in-order delivery of the packets. Hence, Head of Line (HoL) blocking exists at the transport [2] in HTTP/2. Also, the connection establishment latency is higher in TCP by 3-way handshake and with Transport Layer Security (TLS) it is even higher.

To establish the connection faster, Google proposed Quick UDP Internet Connections (QUIC) as an experimental transport protocol in 2013 and standardized by IETF in 2015. QUIC is a multiplexed, connection oriented, reliable and encrypted protocol which operates on top of unreliable User Datagram Protocol (UDP) [3]. QUIC has multiple streams in the transport layer thus eliminating HoL blocking. The data is encrypted at the transport level and it does not induce additional TLS handshake enabling faster connection establishment. While resuming a connection, the data can be sent directly in zero round trip time (0-RTT).

Since there is a transition in the protocols and the streaming experience of the existing ABR algorithms is unexplored with respect to HTTP/3, we aim to bridge the gap and this motivates the study of QoE with HTTP/3 over QUIC. This paper aims to address the following questions:

- What is the impact by switching transport from HTTP/2 to HTTP/3 on ABR performance and QoE?
- Among the existing ABR algorithms, which ABR algorithm is best suited for HTTP/3 over QUIC?

To answer them, an experimental comparison is carried out between HTTP/3 over QUIC and HTTP/2 over TCP for different ABR algorithms ranging from traditional *throughput* to advanced low-latency algorithms (L2A and LoL+) over the public internet. In addition to the QoE, we analyze the details of every segment received to understand the behavior of the ABR algorithms.

The key findings of this paper are,

- With the advent of HTTP/3, a higher QoE is achieved with most ABR algorithms by utilizing higher throughput of QUIC and downloading media segments faster than HTTP/2 over TCP.
- Among the existing ABR algorithms, low latency algorithms (L2A or LoL+) achieve higher bit rate, minimize bitrate switches, and maximize the average throughput with HTTP/3 over QUIC under most network conditions.

II. Background

Various studies have been done to infer the QoE with respect to QUIC. A study presented in [4] evaluated the performance with video streaming and found no improvement in QoE. Several studies of ABR algorithms in DASH streaming compared QUIC and TCP and they find TCP delivers a better QoE [5] [6] [7]. In a follow up work, the authors find that the average bit rate increases with QUIC retransmissions [8] and QUIC starts media streams quickly in congested networks [9]. Since all the data that flows over QUIC is encrypted [10]–[12] propose methodologies to infer QoE from encrypted traffic. Using HTTP/3 over QUIC, the QoE was evaluated by an open source auditing tool Lighthouse [13]. In [14], the adaptability between HTTP/3 over QUIC and DASH video streaming is proposed.

Adaptive bitrate is the standard approach used by streaming servers to adjust delivered bit rate of the media to the underlying playback and network conditions. The video is fragmented into small segments (2 to 10 seconds) and it is encoded into different bit rates and stored in the server. The DASH client has a buffer controller, throughput estimator, ABR controller and scheduler. The buffer controller checks the buffer occupancy on the player. The throughput estimator estimates the throughput on the network. The ABR controller decides the bitrate based on the buffer occupancy and the throughput estimate. The scheduler makes a HTTP request for the segment with the corresponding bitrate and the server sends back the HTTP response for the segment. Several ABR algorithms were designed with the goal of delivering higher video quality and minimizing the stalls. The following ABR algorithms are evaluated as described in Section IV.

Throughput The traditional throughput algorithm estimates the bandwidth of the network using a sliding window or Exponential Weighted Moving Average. The playback bit rate is chosen based on the available bandwidth of the network.

BOLA (Buffer Occupancy based Lyapunov Algorithm) [15] It is a buffer based algorithm which uses Lyapunov Optimization. They derive a utility which aims to provide minimum rebuffering and maximum average bitrate.

Dynamic [16] Dynamic ABR uses throughput based algorithm and buffer based algorithm for selecting bitrates. When buffer levels are low (start up state, seek state) it selects throughput ABR algorithm and when buffer levels are high (steady state) it switches to BOLA. This is the default algorithm used by the standard reference player.

L2A (Learn 2 Adapt) [17] Learn to Adapt is a low latency algorithm which involves Online Convex Optimization for bitrate adaptation. It does not require throughput estimation or assumptions of the network. Instead it uses historic values for bit rate selection.

LoL+ (Low On Latency) [18] The Low on Latency Live streaming algorithm uses an unsupervised learning mechanism known as Self Organizing Map (SOM). Weights are assigned for the SOM module. QoE score, throughput, playback speed and weight vector are taken into account for bitrate selection.

III. Experimental Setup

Fig. 1 outlines the setup used to evaluate the performance of DASH using different ABR algorithms with HTTP/3 over QUIC and HTTP/2 over TCP. To validate the experiment as a real life experience, they were conducted over the public internet with varying network conditions.

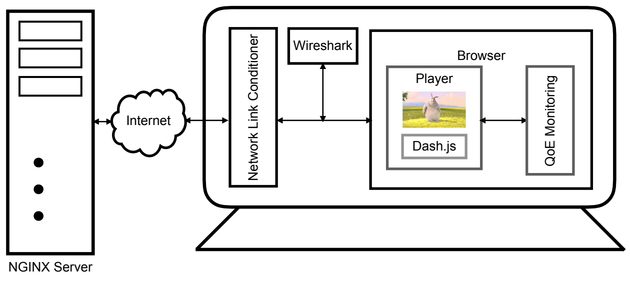


Fig. 1: Experiment Network

A. Experiments

In our experiments, NGINX server (nginx/1.18.0) is used. QUICHE, an implementation of HTTP/3 over QUIC, is integrated into NGINX with an unofficial patch. The NGINX server runs on a Linux 5.10.7 kernel on a Intel (R) Core (TM) i5-6400 CPU @ 2.70GHz processor with 8 GB RAM. NGINX provides support for HTTP/3 and HTTP/2. For the video source a multi codec dash dataset is used [19]. The segments are 2 seconds in length and they are encoded into 19 different bitrates (0.1, 0.2, 0.24, 0.375, 0.55, 0.75, 1, 1.5, 2.3, 3, 4.3, 5.8, 6.5, 7, 7.5, 8, 12, 17, 20 Mbps) and they are stored in the server.

The client runs on macOS BigSur version 11.5 which runs on a 2.2 GHz 6-core Intel core i7 processor with 16 GB RAM. Google Chrome Canary (Version 93.0.4574.0) supports QUIC and HTTP/3 by enabling the flags *–enable-quic* and *–quic-version=h3-29* respectively. After each run the caches are cleared as QUIC uses the information from the cache for 0-RTT. We make sure the connection is terminated as HTTP/3 may send a request over an existing connection without closing them. To switch between various ABR algorithms and to extract video playback metrics like buffer occupancy, segment download time, current playback position and state of the player, we modify *dash.js* v3.2.0. version. The modifications are not significant and should not have an impact on the performance of *dash.js*.

The network traffic is captured using Wireshark. Since the traffic is encrypted in QUIC, the application layer details are not visible. The Secure Sockets Layer (SSL) session keys are obtained to decrypt the traffic to ensure that the traffic goes over HTTP/3 or HTTP/2. Network Link Conditioner is used to throttle the network.

B. Measures

This section outlines the metrics used for obtaining the segment statistics and QoE. To explore the behavior of the ABR algorithms with respect to HTTP/3 and HTTP/2, it is critical to extract the details about the segment (segment statistics). In the following text we assume that the video is fragmented into n segments and the size of each segment is S_i.

We observe a range of segment statistics by obtaining the details about the segment requested to the server and the segment delivered to the client. The buffer occupancy of the player is measured in the client-side by querying the player every 500 ms. The parameter used for analyzing the segment statistics in the client-side, is discussed below

- *Buffer Occupancy*: The buffer occupancy of the player is the difference between the video buffered time and the current playback position. If the buffer is full, the ABR algorithm can request a segment with higher bitrate. If the player's buffer is nearly empty, the ABR algorithm requests for a lower bitrate segment to prevent stalling.

While a segment is requested, the server records the details about the segment in the access log. The following segment statistics metrics are obtained on the server-side by extracting the details from the access log.

- *Segment Download Time (D_i)*: The segment download time D_i is calculated as the difference between the HTTP response time of the segment and the HTTP request time of the segment.
- *Segment Download Throughput (t_i)*: The segment download throughput is calculated as the ratio between segment size S_i and the segment download time D_i

$$t_i = \frac{S_i}{D_i} \tag{1}$$

- *Requested Segment Data Rate*: It represents the bit rate of the segment r requested by the client. The dataset [19] has segments stored in 19 different bitrates. The request for the segment for a particular bitrate is denoted as requested segment data rate.
- *Effective Segment Data Rate (b_r)*: After the client sends a request for the segment r, the server sends the segment r as the response. The bitrate of the segment perceived by the end user is denoted as effective segment data rate.

After collecting the details of the segment statistics, the QoE metrics are calculated, analyzed and discussed below.

- *Average bitrate (B)*: The average bitrate B during the entire playback is calculated as the sum of the individual bitrate of the segments divided by the total number of segments.

$$B = \frac{\sum_{r=i}^{n} b_r}{n} \tag{2}$$

- *Number of bitrate switches*: The number of quality switches is calculated based on the different bitrates rendered with the underlying network conditions. A lower value indicates the steadiness in playback quality.
- *Average Throughput (T)*: Average throughput during the entire playback is calculated by averaging the throughput of individual segments.

$$T = \frac{\sum_{r=i}^{n} t_r}{n} \tag{3}$$

IV. Results

This section presents the segment statistics and QoE / Performance measures for various ABR algorithms under varying network conditions and application protocols. In addition to the normal network conditions, the behavior of the ABR algorithms is studied under different congested network conditions by inducing loss and latency. The network is throttled to 100 Mbps with loss 2% and an additional 10 ms latency is selectively introduced. This leads to three scenarios: BW (bandwidth 100 Mbps), BW-loss (bandwidth 100 Mbps, loss 2%), BW-loss-latency (bandwidth 100 Mbps, loss 2%, latency 10 ms). By varying the network conditions, ABR algorithms and application protocols we run about 30 experiments. For every experiment conducted, we analyze the segment statistics and compare it between different ABR algorithms and network conditions.

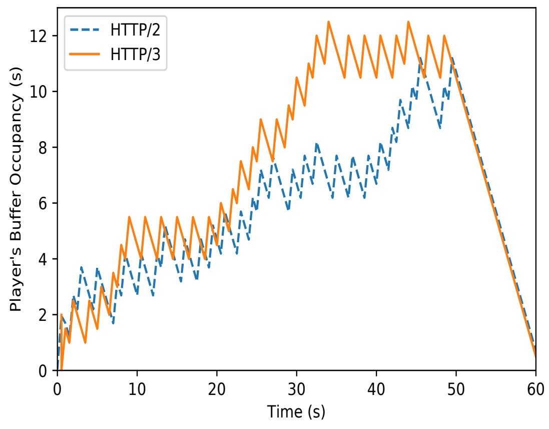
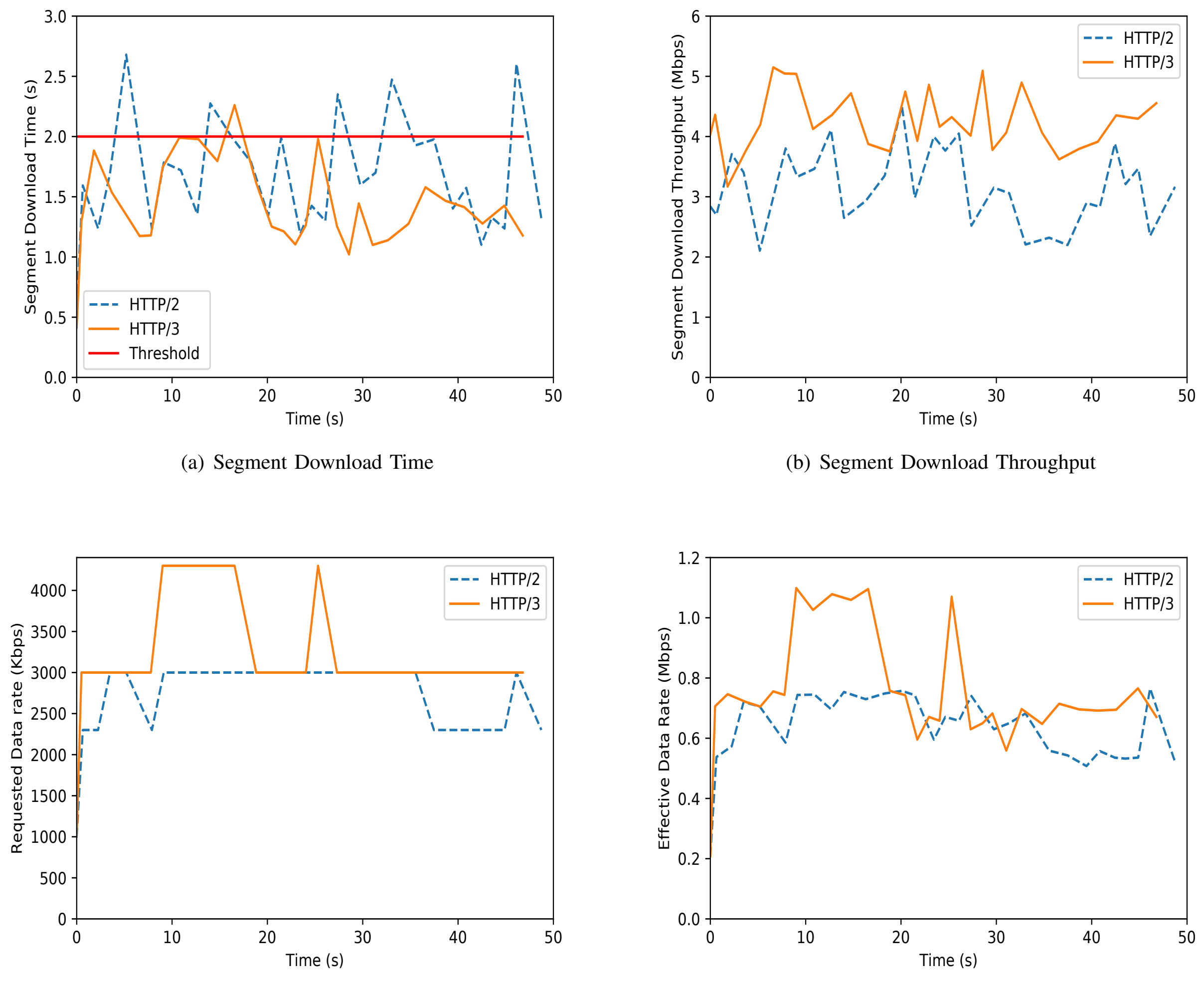


Fig. 2: Client-side measurement - Buffer Occupancy of the player for LoL+ ABR algorithm under the network scenario BW-loss.

A. Segment Statistics

We extract as much as possible information from the client and the server to understand the behavior of the ABR algorithms and the protocols. The segment statistics are collected and compared for all the experiments. Due to the page limitation, we provide the results for LoL+ ABR algorithm under BW-loss. From the client, the buffer occupancy of the player is recorded and represented in Fig. 2. Knowing the buffer occupancy is crucial while deciding the bitrate. If the player has more data in its buffer it can request to switch to higher bitrates. In Fig. 2, HTTP/3 has more buffer occupancy than HTTP/2.



(a) Segment Download Time (b) Segment Download Throughput

(c) Requested Segment Data Rate (d) Effective Segment Data Rate

Fig. 3: Server-side measurements - Detailed experimental results for LoL+ ABR algorithm under the network BW-loss.

The measurements taken from the server are represented in Fig. 3. The time taken by a segment to download is recorded in Fig. 3(a). The segment duration is represented as a threshold. The segments are of 2 seconds duration, so the threshold is 2 seconds. We find that HTTP/3 downloads the segments faster than HTTP/2. Thus, QUIC media streams are downloaded faster. A correspondence is noted between the time taken to download a segment and the segment download throughput.

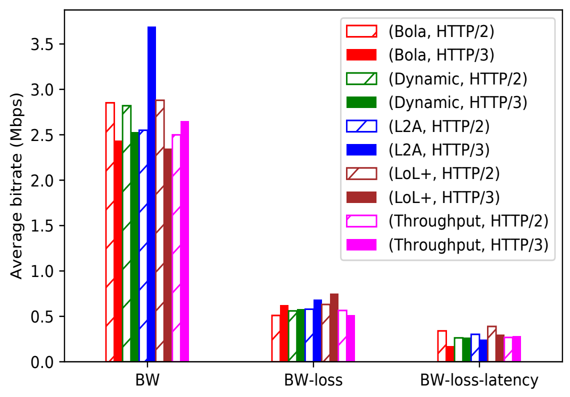
If the segment download time exceeds the threshold, a drop in throughput is noticed in Fig. 3(b). The quicker the segment downloads the higher the throughput is. The segment download throughput is calculated from Eq. (1). In the Fig. 3(b) HTTP/3 has higher throughput over HTTP/2 as the segments were downloaded faster.

Likewise, when a segment takes more time to download and when the player's buffer occupancy is low, a lower data rate is requested for the next segment as in Fig. 3(c). A lower data rate is observed during playback in HTTP/2 as shown in Fig. 3(d). After analyzing the segment statistics we notice that HTTP/3 over QUIC downloads the segments faster than HTTP/2 over TCP in lossy network conditions. In addition to the segment statistics, the QoE parameters are analyzed in the next section.

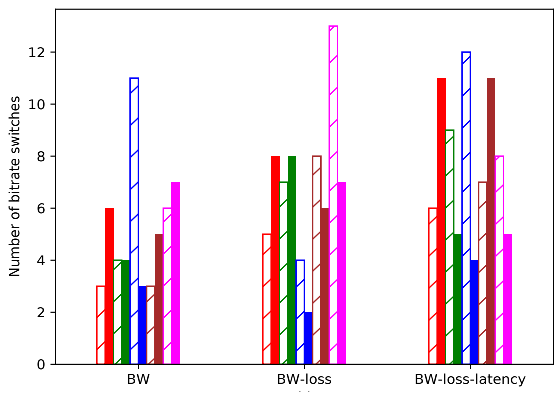
B. *QoE / Performance Metrics*

After collecting the details about the segments, the results are consolidated and represented per network scenario in Fig. 4. This helps in finding the best suited ABR algorithm for HTTP/3 over QUIC.

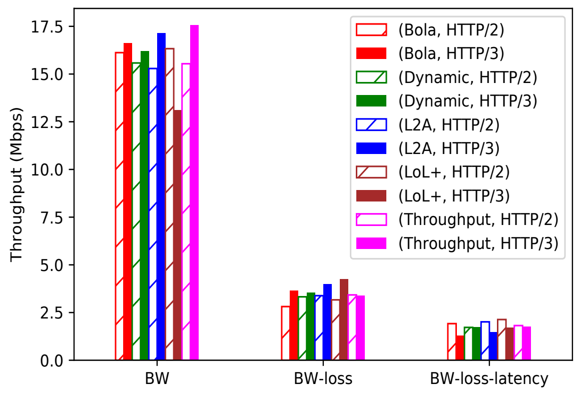
1) Average bitrate: The average bitrate corresponds to the quality of the video streamed. When the network has ample bandwidth and when the playback buffer is almost full, the ABR algorithm requests for a segment with higher bitrate. A higher average bitrate represents that a higher quality video is delivered. The average bitrate B is calculated from Eq. (2) and it is represented in Fig. 4(a).



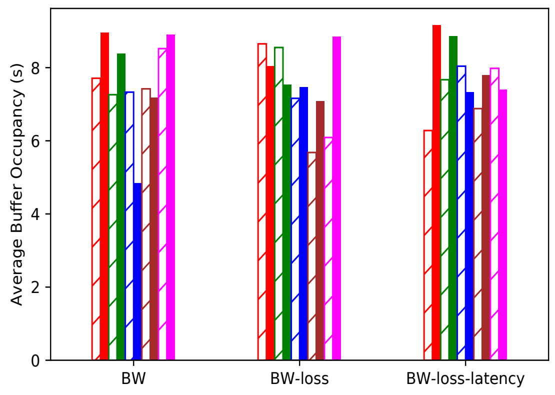
(a) Average Bitrate



(b) Number of Bitrate Switches



(c) Average Throughput



(d) Average Buffer Occupancy

Fig. 4: QoE / Performance metrics under various network conditions, ABR algorithms between HTTP/3 over QUIC and HTTP/2 over TCP.

HTTP/3 over QUIC achieves the maximum average bit rate when L2A algorithm operates under the BW scenario. LoL+ algorithm attains the maximum average bitrate when it operates under BW-loss scenario for HTTP/3 over QUIC and BW-loss-latency scenario for HTTP/2 over TCP.

2) Number of bitrate switches: When the bandwidth of the network is lower or when the playback buffer is about to deplete, the client requests for a lower bitrate to prevent stalling and continue the playback. A fluctuation in the bitrate will decrease the QoE. Fig. 4(b) represents the number of bitrate switches.

In the BW scenario, L2A has minimum bitrate switches while operating under HTTP/3 over QUIC and BOLA, LoL+ have the least bitrate switches while operating under HTTP/2 over TCP. In BW-loss and BW-loss-latency scenarios, L2A has the least bitrate switches with HTTP/3 over QUIC.

On all network conditions, L2A has the least quality fluctuation with HTTP/3 over QUIC.

3) Average throughput: The average throughput is calculated from Eq. (3) for varying network conditions and it is represented in Fig. 4(c). Under the BW scenario, Throughput and L2A algorithms have the maximum average throughput with HTTP/3 over QUIC. Under the BW-loss scenario, LoL+ and L2A attain the highest average throughput with HTTP/3 over QUIC. While operating with a BW-loss-latency scenario, LoL+ and L2A algorithms achieve the maximum average throughput with HTTP/2 over TCP. Predominantly, the low latency algorithms achieve higher average throughput.

4) Average Buffer Occupancy: The average buffer occupancy of the player for various ABR algorithms is represented in Fig. 4(d). When the player's buffer depletes, the risk of stall is higher. Whereas when the player's buffer is full it can switch to a higher bitrate. In the BW scenario, BOLA and Throughput ABR algorithms have the highest average buffer occupancy while operating under HTTP/3 over QUIC. In the BW-loss scenario, Throughput ABR algorithm has the highest average buffer occupancy with HTTP/3 over QUIC. BOLA has the highest average buffer occupancy while operating under the BW-loss-latency scenario with HTTP/3 over QUIC.

TABLE I: The best performing ABR algorithm

Network Impairments	Average bitrate	Number of bitrate switches	Average throughput	Average buffer occupancy
BW	**L2A**	**L2A**, BOLA, LoL+	**Throughput, L2A**	**BOLA, Throughput**
BW-loss	**LoL+**	**L2A**	**LoL+, L2A**	**Throughput**
BW-loss-latency	LoL+	**L2A**	LoL+, L2A	**BOLA**

ABR algorithms running over HTTP/3 are represented in **bold red**. ABR algorithms running over HTTP/2 are represented in black.

C. The best performing ABR algorithms

Table I consolidates the ABR algorithms that outperform, under different network conditions. It can be observed that the low latency algorithms have higher average bitrate, lesser quality fluctuations, higher throughput and lower buffer occupancy with HTTP/3 over QUIC in most scenarios. With respect to different QoE parameters, we notice that the low latency algorithms (L2A and LoL+) work in favor of HTTP/3 over QUIC.

D. Application Layer Protocol

In the existing literature [5]–[7], QUIC exhibits lower QoE when compared to TCP with different ABR algorithms. But with the advent of HTTP/3, our experiments show that HTTP/3 over QUIC achieves higher QoE in many scenarios. As observed in [9] [20], we also notice that HTTP/3 over QUIC has better QoE than HTTP/2 over TCP in lossy network (BW-loss scenario) conditions. We find that, HTTP/3 over QUIC fetches media streams faster and that leads to improved QoE.

V. Conclusions

The advent of HTTP/3 over QUIC imposes a need in revisiting the ABR algorithms in terms of QoE. The ABR algorithms are sensitive to the underlying transport protocol and behave differently when operating over HTTP/3 vs HTTP/2. In this paper, we find that most ABR algorithms achieve higher QoE with HTTP/3 over QUIC in lossy network conditions by quickly downloading the media streams. The low latency algorithms achieve higher bitrate, higher throughput and minimize the number of quality switches with HTTP/3 over QUIC at different network scenarios in majority of the experiments. Among the existing ABR algorithms, we find that the low latency algorithms (L2A and LoL+) have a greater scope with HTTP/3 over QUIC. We conclude that, while streaming with HTTP/3 over QUIC, low latency algorithms could lead to improvement in QoE.

References

[1] CISCO, "Global - 2021 forecast highlights." [Online]. Available: https://www.cisco.com/c/dam/m/en_us/solutions/service-provider/vni-forecast-highlights/pdf/Global_2021_Forecast_Highlights.pdf

[2] R. Marx, "Head-of-line blocking in quic and http/3: The details," 2020. [Online]. Available: https://calendar.perfplanet.com/2020/head-of-line-blocking-in-quic-and-http-3-the-details/#sec_what

[3] A. Langley et al., "The QUIC transport protocol: Design and internet-scale deployment," in *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, ser. SIGCOMM '17. Association for Computing Machinery, 2017, p. 183–196.

[4] M. Seufert, R. Schatz, N. Wehner, and P. Casas, "QUICker or not? -an empirical analysis of QUIC vs TCP for video streaming QoE provisioning," in *2019 22nd Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN)*, 2019, pp. 7–12.

[5] D. Bhat, A. Rizk, and M. Zink, "Not so QUIC: A performance study of DASH over QUIC," in *Proceedings of the 27th Workshop on Network and Operating Systems Support for Digital Audio and Video*, ser. NOSSDAV'17. Association for Computing Machinery, 2017, p. 13–18.

[6] A. Mondal and S. Chakraborty, "Does QUIC suit well with modern adaptive bitrate streaming techniques?" *IEEE Networking Letters*, vol. 2, no. 2, pp. 85–89, 2020.

[7] S. Arisu et al., "Game of protocols: Is QUIC ready for prime time streaming?" *Int. J. Netw. Manag.*, vol. 30, no. 3, May 2020.

[8] D. Bhat, R. Deshmukh, and M. Zink, "Improving QoE of ABR streaming sessions through QUIC retransmissions," in *Proceedings of the 26th ACM International Conference on Multimedia*, ser. MM '18. Association for Computing Machinery, 2018, p. 1616–1624.

[9] S. Arisu and A. C. Begen, "Quickly starting media streams using QUIC," in *Proceedings of the 23rd Packet Video Workshop*, ser. PV '18. Association for Computing Machinery, 2018, p. 1–6.

[10] Tisa-Selma, A. Bentaleb, and S. Harous, "Inferring quality of experience for adaptive video streaming over HTTPS and QUIC," in *2020 International Wireless Communications and Mobile Computing (IWCMC)*, 2020, pp. 81–87.

[11] M. H. Mazhar and Z. Shafiq, "Real-time video quality of experience monitoring for HTTPS and QUIC," in *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications*, 2018, pp. 1331–1339.

[12] S. Xu, S. Sen, and Z. M. Mao, "CSI: Inferring mobile ABR video adaptation behavior under HTTPS and QUIC," in *Proceedings of the Fifteenth European Conference on Computer Systems*, ser. EuroSys '20. Association for Computing Machinery, 2020.

[13] D. Saif, C.-H. Lung, and A. Matrawy, "An early benchmark of quality of experience between HTTP/2 and HTTP/3 using lighthouse," 2020.

[14] S. Chellappa and R. Bartos, "Adaptability between abr algorithms in dash video streaming and http/3 over quic: Research proposal," in *Proceedings of the 13th ACM Multimedia Systems Conference*, ser. MMSys '22. Association for Computing Machinery, 2022, p. 388–392. [Online]. Available: https://doi.org/10.1145/3524273.3533932

[15] K. Spiteri et al., "BOLA: near-optimal bitrate adaptation for online videos," in *IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications*, 2016, pp. 1–9.

[16] ——, "From theory to practice: Improving bitrate adaptation in the DASH reference player," *ACM Trans. Multimedia Comput. Commun. Appl.*, vol. 15, no. 2s, Jul. 2019.

[17] T. Karagkioules et al., "Online learning for low-latency adaptive streaming," in *Proceedings of the 11th ACM Multimedia Systems Conference*, ser. MMSys '20. Association for Computing Machinery, 2020, p. 315–320.

[18] M. Lim et al., "When they go high, we go low: Low-latency live streaming in dash.js with lol," in *Proceedings of the 11th ACM Multimedia Systems Conference*, ser. MMSys '20. Association for Computing Machinery, 2020, p. 321–326. [Online]. Available: https://doi.org/10.1145/3339825.3397043

[19] A. Zabrovskiy et.al, "Multi-codec DASH dataset," in *Proceedings of the 9th ACM Multimedia Systems Conference*, ser. MMSys '18. Association for Computing Machinery, 2018, p. 438–443.

[20] A. Kakhkiet al., "Taking a long look at quic: An approach for rigorous evaluation of rapidly evolving transport protocols," in *Proceedings of the 2017 Internet Measurement Conference*, ser. IMC '17. Association for Computing Machinery, 2017, p. 290–303. [Online]. Available: https://doi.org/10.1145/3131365.3131368